R examples for fitting GLMS

O-ring Data

We will illustrate basic features of fitting GLMs in R. To begin, consider the binary version of the O-ring data, where the response is whether 1 or more O-rings failed during flight. We consider two predictors, temperature at lift-off and O-ring pressure. I stored the data as a text file, in a "rectangular array" format – a row for each flight, and separate columns for the variables. The file had a "header" with labels for each column of data.

After entering R, we read and print the data to the R console. The variable labels refer to flight number, the binary response of at least 1 O-ring failure, how many of 6 failed, and then temperature and pressure.

```
> aa = read.table("D:/My Documents/GLMcourse/glmSECTION/orings.tex",header=T)
> aa

Flight Resp Nof6 Temp Pressure
1     14   1   2   53      50
2      9   1   1   57      50
3     23   1   1   58     200
4     10   1   1   63      50
5      1   0   0   66     200
6      5   0   0   67      50
7     13   0   0   67     200
8     15   0   0   67      50
9      4   0   0   68     200
10     3   0   0   69     200
11     8   0   0   70      50
12    17   0   0  70     200
13     2   1   1  70     200
14    11   1   1  70     200
15     6   0   0  72     200
16     7   0   0  73     200
17    16   0   0  75     100
18    21   1   2  75     200
19    19   0   0  76     200
20    22   0   0  76     200
21    12   0   0  78     200
22    20   0   0  79     200
23    18   0   0  81     200
```

The data is formatted as a data frame, a standard R format for model fitting. To get help on the structure of the read.table command, just type

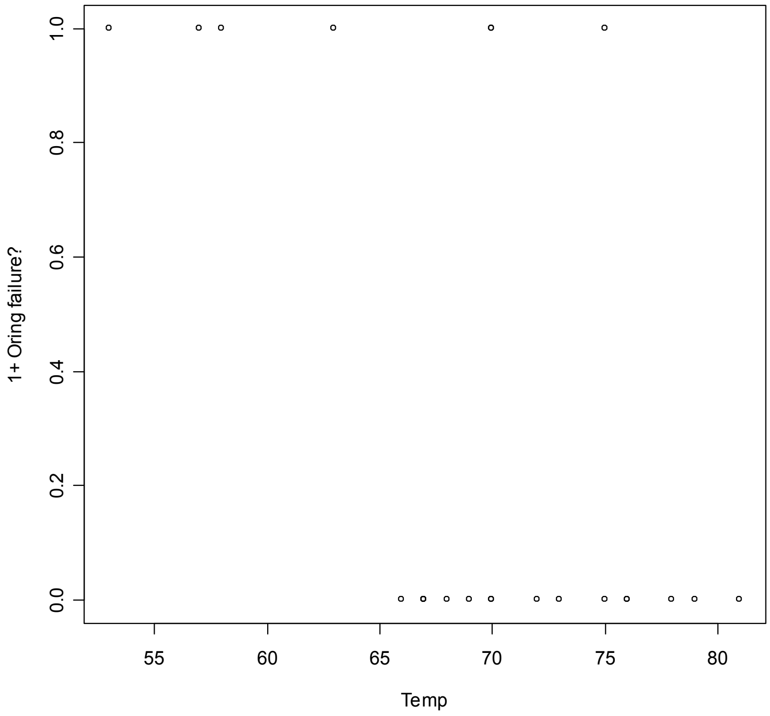
```
> help(read.table)
```

Documentation will appear in a new window. Reading through the help you will recognize that there are several format in which data can be stored, for example comma or tab delimited.

A data frame is a "list" which has components. To reference columns of aa type aa$Resp, aa$Temp, etc. Alternatively, you could assign these to new variables.

Here is a simple plot of the response as a function of temperature:

```
>> plot(aa$Temp,aa$Resp,xlab="Temp",ylab="1+ Oring failure?")
```



To fit a logistic regression model with Temperature and Pressure as predictors, enter the command:

```
> a1 <- glm(Resp ~ Temp + Pressure, family= binomial, data=aa)
```

This creates a linear model class object with many components. The input to glm is a model statement, the family of distributions (for the binomial, the logit link is default), and the data frame that contains variables referenced in the model.

Selected commands can be used on the created object a1 to produce output. For example, the summary command provides a parameter estimates table and deviance, and information on the model that was fitted:

```
> summary(a1)

Call: glm(formula = Resp ~ Temp + Pressure, family = binomial, data = aa)

Deviance Residuals:

   Min       1Q     Median      3Q      Max
-1.1928  -0.7879  -0.3789   0.4172   2.2031

Coefficients:
           Estimate         Std. Error   z value  Pr(>|z|)
(Intercept) 16.38531     8.02747    2.041    0.0412 *
Temp         -0.2634      0.12637   -2.084    0.0371 *
Pressure      0.0051      0.00925    0.559    0.5760

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

   Null deviance:    28.267  on 22  degrees of freedom
Residual deviance: 19.984  on 20  degrees of freedom
AIC: 25.984

Number of Fisher Scoring iterations: 5
```

The null deviance is the deviance from a model with an intercept only, so adding the 2 predictors decreases the deviance by approximately 8.28. Note that the pressure effect is not significant at any of the usual significance levels. The z-values and corresponding p-values correspond to Wald tests.

The anova command provides a sequential Analysis of Deviance table with the sequential reduction in deviance achieved by adding predictors in the order specified on the model statement (first Temp then Pressure). Can everybody figure out the output?

```
> anova(a1)

Analysis of Deviance Table

Model: binomial, link: logit
Response: Resp
Terms added sequentially (first to last)

            Df       Deviance    Resid. Df    Resid. Dev
NULL                                22            28.2672
Temp         1          7.9520       21            20.3152
```

```
Pressure     1         0.3314         20              19.9838
```

The names command identifies components of the output object a1 that are available for printing or plotting:

```
> names(a1)

"coefficients"    "residuals"    "fitted.values"   "effects"     "R"    "rank"           "qr"
"family"          "linear.predictors"   "deviance"      "aic"           "null.deviance"     "iter"
"weights"         "prior.weights"    "df.residual"     "df.null"         "y"    " converged"
"boundary"     "model"      "call"   "formula"      "terms"       "data"      "offset"
"control"        "method"            "contrasts"        "xlevels"
```

To get more information on these, ask for help on glm.

Here are the (first few) fitted values, “working residuals” (will describe in words) and the estimated linear predictors. There are other ways to get fitted values and residuals – see next example.

```
> a1$fitted.values
0.93606295 0.83619258 0.89505077 0.51243319 0.50904180

> a1$residuals
 1.068296  1.195892  1.117249  1.951474 -2.036833 -1.366457

> a1$linear.predictors
 2.68378374  1.63016745  2.14340375  0.04974301  0.03617116
```

The update command is convenient for deleting or adding predictors to a model which has already been fitted and output saved in a linear model object. To drop Pressure from our logistic model, specify:

```
> a1new <- update(a1, . ~ . - Pressure)
> summary(a1new)

Call:
glm(formula = Resp ~ Temp, family = binomial, data = aa)

Deviance Residuals:
   Min       1Q      Median       3Q      Max
-1.0611  -0.7613  -0.3783   0.4524   2.2175

Coefficients:
              Estimate     Std. Error z value   Pr(>|z|)
(Intercept)  15.0429     7.3786       2.039   0.0415 *
Temp         -0.2322        0.108     -2.145   0.0320 *
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance:    28.267  on 22  degrees of freedom
Residual deviance: 20.315  on 21  degrees of freedom
AIC: 24.315

Number of Fisher Scoring iterations: 5
```

To add pressure back into the model (some output omitted):

```
> summary( update(a1new, . ~ . + Pressure) )

Call:
glm(formula = Resp ~ Temp + Pressure, family = binomial, data = aa)

Coefficients:
            Estimate      Std. Error    z value Pr(>|z|)
(Intercept) 16.3853    8.027474   2.041    0.0412 *
Temp        -0.2634    0.126371  -2.084    0.0371 *
Pressure      0.0051    0.009257   0.559     0.5760

 (Dispersion parameter for binomial family taken to be 1)
```

Further information on specifying link functions for various GLMs is obtained via

```
> help(family)

family                package:stats                 R Documentation

Family Objects for Models
Description:

     Family objects provide a convenient way to specify the details of
     the models used by functions such as 'glm'.  See the documentation
     for 'glm' for the details on how such model fitting takes place.

Usage:

     family(object, ...)

     binomial(link = "logit")
     gaussian(link = "identity")
     Gamma(link = "inverse")
     inverse.gaussian(link = "1/mu^2")
     poisson(link = "log")
     quasi(link = "identity", variance = "constant")
     quasibinomial(link = "logit")
     quasipoisson(link = "log")
```

Arguments:

link: a specification for the model link function. The 'gaussian' family accepts the links '"identity"', '"log"' and '"inverse"'; the 'binomial' family the links '"logit"', '"probit"', "cauchit"', (corresponding to logistic, normal and Cauchy CDFs respectively) '"log"' and '"cloglog"' (complementary log-log); the 'Gamma' family the links '"inverse"', '"identity"' and '"log"'; the 'poisson' family the links '"log"', '"identity"', and '"sqrt"' and the 'inverse.gaussian' family the links '"1/mu^2"', '"inverse"', '"identity"' and '"log"'.

For example, a probit fit is obtained via:

```
> a2 <- glm(Resp ~ Temp + Pressure, family= binomial(link=probit), data=aa)
> summary(a2)

Call:

glm(formula = Resp ~ Temp + Pressure, family = binomial(link = probit), data = aa)

Deviance Residuals:

    Min       1Q   Median       3Q      Max
-1.2202  -0.7915  -0.3737   0.3978   2.1934

Coefficients
              Estimate   Std. Error  z value  Pr(>|z|)
(Intercept)  9.647729   4.197527    2.298   0.0215 *
Temp        -0.155793   0.066325   -2.349   0.0188 *
Pressure     0.003487   0.005374   0.649   0.5165

(Dispersion parameter for binomial family taken to be 1)

    Null deviance:    28.267  on 22  degrees of freedom
Residual deviance: 19.977  on 20  degrees of freedom
AIC: 25.977

Number of Fisher Scoring iterations: 6
> anova(a2)

Analysis of Deviance Table
Model: binomial, link: probit
```

```
Response: Resp

Terms added sequentially (first to last)

           Df  Deviance   Resid. Df   Resid. Dev
NULL                          22          28.2672
Temp      1   7.8894         21          20.3777
Pressure  1   0.4012         20          19.9765
```

For binomial data with samples sizes not all equal to 1, the response has to be captured in two columns, one for the successes and the other for failures. The cbind function can be used when successes and failures are two columns of the data frame, or when, as here, the sample sizes are the same. Note that summaries do not change

```
> a4 <- glm( cbind(Resp,1-Resp) ~ Temp + Pressure, family= binomial(link=probit),
data=aa)
> summary(a4)

Call:
glm(formula = cbind(Resp, 1 - Resp) ~ Temp + Pressure, family = binomial(link =
probit),     data = aa)

Coefficients:
              Estimate    Std. Error   z value Pr(>|z|)
(Intercept)  9.647729   4.197527   2.298   0.0215 *
Temp        -0.155793   0.066325  -2.349   0.0188 *
Pressure     0.003487   0.005374   0.649   0.5165

 (Dispersion parameter for binomial family taken to be 1)

   Null deviance:    28.267  on 22  degrees of freedom
Residual deviance: 19.977  on 20  degrees of freedom
AIC: 25.977
```

Here are summaries from the complementary log-log fit:

```
> a3 <- glm(Resp ~ Temp + Pressure, family= binomial(link=cloglog), data=aa)
> summary(a3)

Call:
glm(formula = Resp ~ Temp + Pressure, family = binomial(link = cloglog), data = aa)

Deviance Residuals:
   Min       1Q   Median      3Q      Max
-1.1279  -0.7751  -0.3942   0.1605   2.1874
```

```
Coefficients:
              Estimate    Std. Error  z value Pr(>|z|)
(Intercept) 12.820913   5.458177   2.349    0.0188 *
Temp        -0.210264   0.087812  -2.394    0.0166 *
Pressure     0.003020   0.006926   0.436     0.6628

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance:    28.267  on 22  degrees of freedom
Residual deviance: 19.336  on 20  degrees of freedom
AIC: 25.336

> anova(a3)

Analysis of Deviance Table
Model: binomial, link: cloglog
Response: Resp

Terms added sequentially (first to last)

           Df   Deviance         Resid. Df    Resid. Dev
NULL                                22            28.2672
Temp      1     8.7357              21              19.5315
Pressure  1    0.1959              20             19.3355
```

A reasonable concern would be how to compare the fits from the various link functions? Ignoring the fact that we would likely omit Pressure from each model, we can compare the deviances, or alternatively the AIC (Akaike Information Criterion: minus twice the maximized log likelihood plus twice the number of parameters). Smaller values of the Deviance or AIC are preferred. Because each of the 3 models has the same number of parameters, the ordering among models based on either the Deviance or AIC are identical.

Link	AIC	Deviance
Logit	25.98	19.98
Probit	25.98	19.98
Complementary log-log	25.34	19.34

The differences in Deviances and AICs among links are small, but a formal selection would choose the complementary log-log link. In practice, it is also useful to compare the observed and fitted proportions, and to do a diagnostic analysis before settling on one of these three links. Here, a comparison of the observed and fitted proportions is not very fruitful – why?

Now let us fit the more refined model that uses Nof6 as response.
Here we need to use the “weights” option to code the number of trials in each flight (n_j=6).

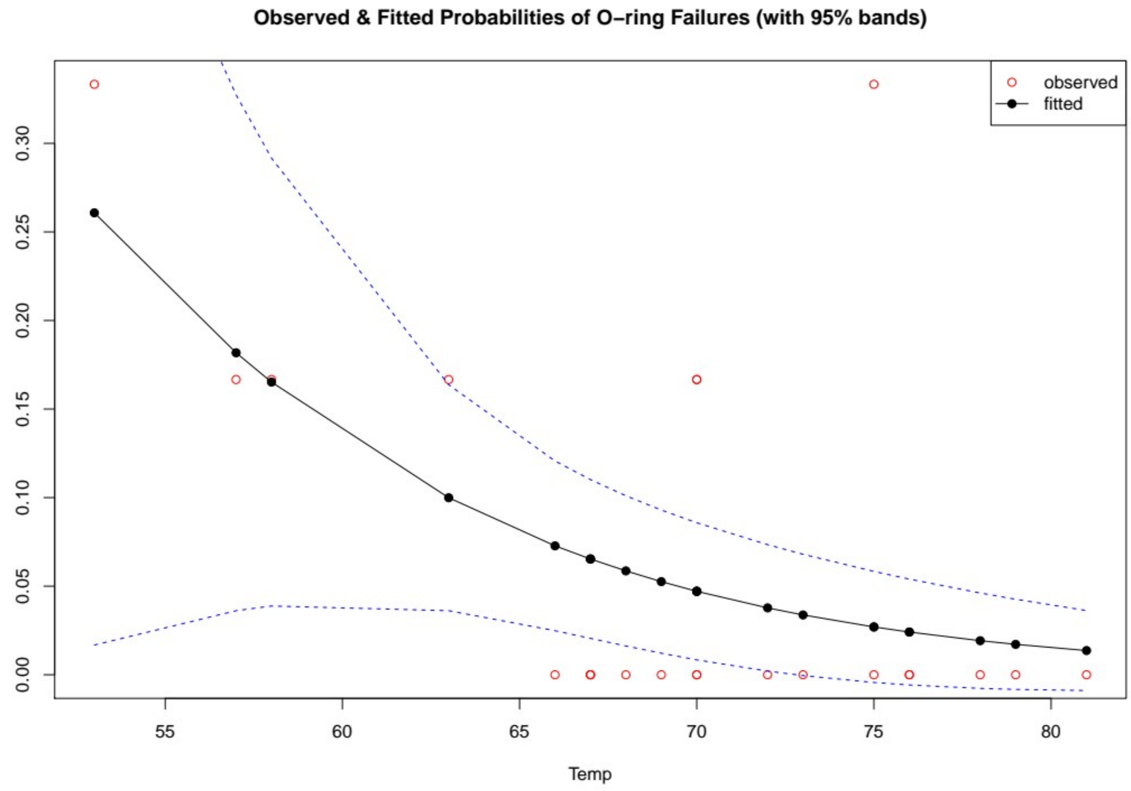
```
> b1 <- glm((Nof6/6) ~ Temp + Pressure, weights=rep(6,23),
family= binomial,      data=orings)
> summary(b1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0113  -0.8024  -0.5436  -0.1031   2.6373

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.548583   3.305918   1.678   0.0933 .
Temp        -0.127227   0.056792  -2.240   0.0251 *
Pressure     0.002144   0.005809   0.369   0.7120
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 24.230  on 22  degrees of freedom
Residual deviance: 17.949  on 20  degrees of freedom
AIC: 37.509
```

Update by dropping Pressure.

```
b1new <- update(b1, . ~ . - Pressure)
summary(b1new)
Coefficients:
          Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.08498    3.05247   1.666   0.0957 .
Temp        -0.11560    0.04702  -2.458   0.0140 *
---
Null deviance: 24.230  on 22  degrees of freedom
Residual deviance: 18.086  on 21  degrees of freedom
AIC: 35.647
```

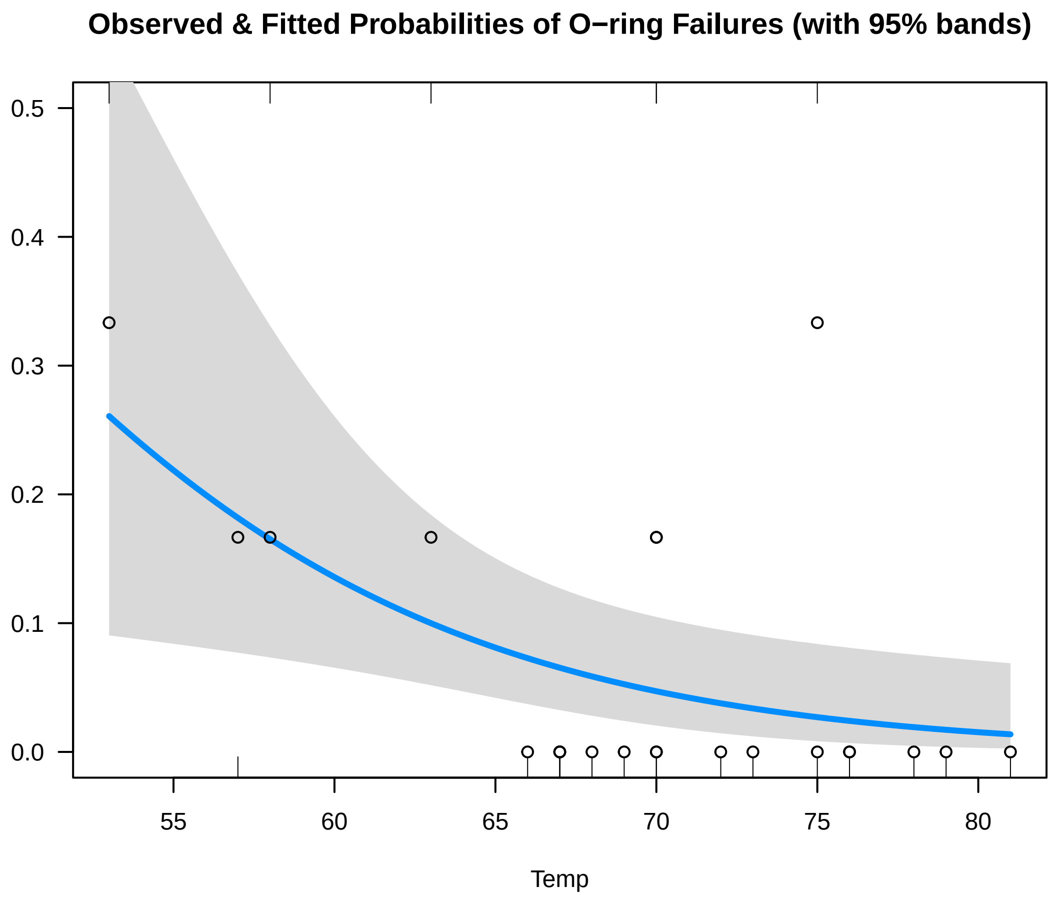
Plot fitted & observed probs vs. the covariate (Temp), with 95% confidence band. Model does not seem to fit too well ... perhaps try other links, more covariates ...

```
p.fit=b1new$fitted; p.obs=(orings$Nof6/rep(6,23));
p.fit.se=predict(b1new, newdata=NULL, type="response", se.fit=TRUE)
$se.fit
p.fit.lower95=p.fit-1.96*p.fit.se;
p.fit.upper95=p.fit+1.96*p.fit.se;
plot(orings$Temp, p.obs, xlab="Temp", ylab="", col="red")
title("Observed & Fitted Probabilities of O-ring Failures (with 95%
bands)")
points(orings$Temp, p.fit, pch=19)
lines(orings$Temp, p.fit, lty=1)
lines(orings$Temp, p.fit.lower95, lty=2, col="blue")
lines(orings$Temp, p.fit.upper95, lty=2, col="blue")
legend("topright", legend=c("observed","fitted"), ,
col=c("red","black"), pch=c(1,19), lty=c(0,1), cex=1)
```



Package “visreg” has a nice canned way of plotting this:

```
library(visreg)
vis1=visreg(b1new, xvar="Temp", scale="response", plot=FALSE)
plot(vis1, xlab="Temp", ylab="", main="Observed & Fitted
Probabilities of O-ring Failures (with 95% bands)", ylim=c(0,0.5))
points(orings$Temp,p.obs)
```


Observed & Fitted Probabilities of O−ring Failures (with 95% bands)
Temp